The User Manual

August 14, 2015

Beijing LongXun Quantum Technology Co., Ltd.

INDEX

1. Introduction 1
2. Input files 5
2.1 etot.input 5
2.2 atom.config 27
2.3 Pseudo-potential files: atom.UPF 30
2.4 IN.KPT 30
2.5 IN.SYMM 32
2.6 Other input files 33
3. Output files 35
3.1 Standard output file 35
3.2 REPORT 35
3.3 RELAXSTEPS 38
3.4 NEB.BARRIER 40
3.5 MDSTEPS 41
3.6 MOVEMENT 42
3.7 Other output files 42
4. The basic calculations 44
4.1 Self-consistent calculations (JOB=SCF) 44
4.2 None self-consist calculations (JOB=NONSCF) 44
4.3 Density of States Calculations (JOB=DOS) 46
4.4 Atomic Relaxation (JOB=RELAX) 47
4.5 Nudged Elastic Band Calculations (JOB=NEB) 48
4.6 Molecular Dynamics (JOB=MD) 49
4.7 Compatible runs with PWSCF 50
5. Pre- and post-processing programs 52
5.1 Pre-processing 52
5.1.1 convert_to_config.x 52
5.1.2 check.x 52
5.2 Post-processing 53
5.2.1 convert_from_config.x 53
5.2.2 plot_band_structure.x 54
5.2.3 plot_dos.x 54
5.2.4 convert_rho.x 55
Appendix: A 56
Appendix: B 57

1. Introduction

PWmat package is a plane wave pseudopotential package for density functional theory (DFT) calculations. It can perform the following calculations (indicated by the "JOB" flag in etot.input): SCF (self-consistent-field calculation); NONSCF (non-self-consistent-field calculations, e.g. for band structure calculation); DOS (density of state calculation, which is usually after a NONSCF or SCF calculation); RELAX (atomic relaxation calculation); MD (ab initio molecular dynamics calculation); and NEB (nudged elastic band calculation for barrier heights). PWmat can be run on CPU or CPU/GPU processors. In the GPU run, one CPU is bundled with one GPU. The package also come with a well-tested pseudopotential library in the upf (unified pseudopotential format) for norm-conserving pseudopotentials (atom.ncpp.upf) and ultrasoft pseudopotentials (atom.uspp.upf). PWmat is fully compatible with the popular open source code "Quantum Espresso" module pwscf (with exchangeable data files), but it can be an order of magnitude faster than pwscf since it runs on GPU.

In order to run PWmat, one needs to provide the following input files in the running directory: etot.input (the PWmat control file); atom.config (the atomic position file); atom.upf (pseudopotential files, they can be copied from the provided pseudopotential library).

Optionally, one might also need to provide IN.KPT (k-point file), IN.SYMM (symmetry file), IN.VR (potential input file), IN.WG (wave function input file) and IN.RHO (charge density input file). These files can either be generated from the pre-processing code (check.x, e.g., for IN.KPT, and IN.SYMM), or from previous calculations (IN.VR, IN.WG, IN.RHO).

There are three steps to use PWmat:

(1) Prepare atom.config, and etot.input, and copy atom.upf;

(2) Run PWmat;

(3) Post-process, visualization.

We provide suggestions for how to generate atom.config, e.g., through open-source visualization tools. One can also write atom.config by hand. The etot.input (which tells PWmat what to do) can be very simple, consisted with only a few lines (e.g., 4 lines, see below) while other PW calculation parameters will be provided by default. However, by running check.x, it also generates a long version: etot.input.long, which can be copied and used as etot.input with manual changed parameters for advanced users.

---

2 2
IN.ATOM = atom.config
JOB = SCF
IN.PSP1 = Si.ncpp.upf

We suggest the following specific steps to run PWmat, from the beginning to the end.

(1) Prepare atomic position file atom.config from online database, or/and visualization packages (e.g., VESTA; Avogadro). The output system file format from these packages could be: system.xyz, system.xsf, or system.vasp.

(2) Convert the system.xyz, system.xsf, system.vasp format into atom.config by running our serial utility code: “>convert_to_config.x < system.xyz(xsf, vasp)”. It will generate atom.config file. Note, atom.config can also be prepared by hand, or

user's own software.

(3) Prepare etot.input (see detail explanation in the next section).

(4) Pre-process by running: ">check.x". This will tell you whether there is any error or inconsistency in etot.input. It will also generate the real space grid: n1, n2, n3. This will help you to decide the: "node1, node2" (the first line in etot.input). One needs to make sure that n1*n2 can be divided by node1. Thus one might need to change node1 in etot.input according to n1, n2, n3, or increase n1, n2, n3. Note, the total number of CPU/GPU equals nodes1*nodes2. check.x can be used to generate IN.KPT and IN.SYMM from Monkhorst-Pack parameters "MP_N123=nk1, nk2, nk3, sk1, sk2, sk3" in etot.input. At the same time one needs to add IN.SYMM = T, IN.KPT = T in etot.input in order to use these k-points, and symmetry files. Check.x can also be used to generate IN.KPT for band structure calculations when "KP_INV= M" is used, here M are the number of K-points between two special k-points (which is determined by check.x following the symmetry of the supercell shape).

(5) Run PWmat, e.g., ">mpirun –np num PWmat" (or just ">PWmat > out &" when there is only on GPU in the machine). Note, num, the number of GPU/CPU processors, must be equal to node1*node2. In our code, one CPU is bundled with one GPU.

(6) Post-processing, e.g., ">convert_from_config.x < MOVEMENT" or "> convert_from_config.x < atom.config", will change the .config, or MOVEMENT file to .xyz, and .xsf file for visualization. One can also run: ">convert_rho.x OUT.RHO" (note, NO "<" between convert_rho.x and OUT.RHO) to generate RHO.xsf for isosurface plot of the charge density).

(7) Visualize the structure, or make an animation with the .xyz and .xsf file, using: VMD to watch the animation, and VEST to view one configuration.

For Mstation users, all the above programs are pre-installed in the Mstation.

As discussed above, besides the main code PWmat, we also provide many utility codes (check.x, convert_to_config.x, convert_from_config.x, etc). In the online page (http://www.pwmat.com/pwmat_tutorial), we provide detail tutorials for how to use a few open-source visualization software packages, both for preparing the input files (pre-processing), and analyzing and viewing the results (post-processing). (Note, due to rapid development, there could be some minor changes for the tutorial, e.g., the pseudopotential file should be atom.upf instead atom.vwr, etc. But the basic steps and ideas are the same, and the changes should be obvious). Thus, we provide an easy-to-follow graphically based workflow from the beginning to the end for doing actually research using PWmat. Although these visualization tools are open-sourced, we provide the utilities to link them together and to make them work together seamlessly. Please check the workflow in Appendix A.

2. Input files

PWmat needs a few basic input files to start the calculation: atom.config, etot.input, and atom.upf (pseudopotential files). While the pseudopotential files can be copied from the library for the required atoms, atom.config and etot.input have to be prepared specifically for the calculation. In the following, we explain the etot.input and atom.config files.

2.1 etot.input

The etot.input is the most essential file controlling how to run PWmat, and what are the names of the other input files. Here is an example of a short (essential) etot.input file.

```
-------------------------------------------------------------
2     2
IN.ATOM =     atom.config
JOB     =     SCF
IN.PSP1    =    Si.upf
-------------------------------------------------------------
```

The first two numbers are “node1, node2”. Node1 is the number of processors for plane wave parallelization. Node1 should be able to divide n1*n2 (n1,n2,n3 are FFT grid in real space). Node2 is the number of processor groups for k-point parallelization. It will divide the number of k-points into node2 groups. IN.ATOM indicates the atomic position file (atom.config) (as an PWmat/etot.input convention, all the input file tags will have the form: IN.XXX, while all the output file tags will have the form OUT.XXX). JOB tells the PWmat what kind of jobs to run. IN.PSP1 indicates the name of pseudopotential file of first atom type (the second atom type

will be IN.PSP2). The etot.input has a format of “tag_name = xxx_value”. The orders of different tags can be changed, except the first line “node1, node2”. The names of the variables are case insensitive. Line which starts with # will be an annotation line. After the xxx_value, one can add annotations in the same line as well, e.g., “MP_N123=4,4,4,0,0,0 ! Monkhorst-Pack parameter without shift”.

If one runs “>check.x”, a long version of etot.input: “etot.input.long” will be generated. This etot.input.long can be copied as etot.input, used as the input file. This etot.input.long gives an explicit form for all the parameters to be used in the calculation (generated by defaults). An experienced user can change some of the parameters according to the need. However, one can also just use the short version etot.input to run PWmat, using the default values for most parameters. Note, when running PWmat, it will generate file: “REPORT”. The first part of the REPORT file is the same as the etot.input.long. So, one can also copy this REPORT to etot.input, delete the others lines, use it as an explicit input file.

One example of long etot.input file looks like this:

---

4 2 ! (node1, node2)
IN.ATOM = atom.config
JOB = SCF
IN.PSP1 = Al.upf
Ecut = 20.0000000000000
Ecut2 = 40.0000000000000
Ecut2L = 40.0000000000000
N123 = 16 16 16
NS123 = 16 16 16
N123L = 16 16 16
SPIN = 1
XCFUNTIONAL = 0.000000000000000E+000
Num_Electron = 12.0000000000000
Num_band = 16
WG_ERROR = 1.000000000000000E-004

```
E_Error          =      8.000000000000000E-007
w_cg             =      7.000000000000001E-002
w_scf            =      0.000000000000000E+00
SCF_ITER0        =      20        4
ALGORITHM0       =       3        0.0000        0.10000        1
                         3        0.0000        0.10000        1
                         3        0.0000        0.10000        1
                         3        0.0000        0.10000        1
                        ……
                         3        1.0000        0.10000        1
                         3        1.0000        0.10000        1
                         3        1.0000        0.10000        1
SCF_ITER1        =      6        4
ALGORITHM1       =      3        1.0000        0.10000        1
                        3        1.0000        0.10000        1
                        3        1.0000        0.10000        1
                        3        1.0000        0.10000        1
                        3        1.0000        0.10000        1
                        3        1.0000        0.10000        1
NONLOCAL         =      2
Rcut             =      3.20000004768372
-------------------------------------------------------------
```

In the following, we explain the meaning of each variable in the long version of etot.input. The Variable in red are the mandatory variables, and all the others are optional (will be set automatically by default values).

Variable	Specification
Node1	The number of processors to divide the G-space sphere and n1*n2*n3 grid. Node1 must evenly divides n1*n2 (which will be provided by check.x).
Node2	The number of processor groups to divide the k-points. The best is to have node2 evenly (or almost evenly) divide the number of k-points. Note, the total number of processors

	must equal to: node1*node2
IN.ATOM	**IN.ATOM = atom.config** The atomic positions file. Its specification is described in the section: “atom.config” of this manual.
IN.PSP1/2…	The names of the pseudopotential files. Such as **IN.PSP1 = Na.ncpp.upf** ! for norm conserving **IN.PSP1 = Na.uspp.upf** ! for ultra-soft Currently, different types of pseudopotentials cannot be mixed (i.e, either all atoms are in norm-conserving PSP, or all in ultra-soft). The PWmat can use norm-conserving or ultrasoft pseudopotentials. We only read upf formatted pseudopotential files. UPF format is the Quantum Espresso format. We provide a tested package of pseudopotentials. One can also copy pseudopotential files from the Quantum Espresso website, and use it as it is.
JOB	Controls what PWmat will do. **JOB = SCF**, do only the self-consistent field iterations, self-consistently determine the charge density, will output the total energy. It will not move atoms. **JOB = NONSCF**, do non-self-consistent calculations. It usually inputs a converged potential or charge density, only calculates the eigen wave functions non-self-consistently. It will not calculate total energy, but it can be used to calculate band structure or density of state.

	JOB = RELAX, do atomic position relaxations using DFT force and total energy. Each atomic relaxation step will do one SCF calculation. Might want to have "RELAX_DETAIL" in the etot.input. See below. A concise result will be reported in RELAXSTEPS. The atomic movements for each relaxation step are reported in MOVEMENT. **JOB = NEB**, calculate the barrier height using nudged elastic band method. Must have a variable "NEB_DETAIL" in etot.input file. Besides IN.ATOM, which gives the first valley site atomic position, there must be a second valley site position given in the NEB_DETAIL line. One must precalculate (e.g., using JOB=RELAX) the atomic configuration of these two valley sites before using JOB=NEB to calculate their barrier. See NEB_DETAIL for more details. Output files: RELAXSTEPS, NEB.BARRIER, MOVEMENT. NEB.BARRIER gives the barrier height information. **JOB = DOS**, do density of state (DOS) calculations based on wave function and eigen energy input from previous calculations (must have IN.WG=T, which is also default, and EIGEN_ALL, from previous step calculations). Output files: DOS.totalspin, bpsiiofil10000x. It takes the converged wave function IN.WG, do an atomic orbital projection in order to get the partial DOS. **JOB = MD**, do molecular dynamics (MD) simulations. Must have variable "MD_DETAIL" in etot.input (see below). A concise output is reported in MDSTEPS. The atomic movements for every step are reported in MOVEMENT.

MD_DETAIL	**MD_DETAIL = MD Mstep dt Temp1 Temp2** Note: this is a required line for JOB=MD. There is no default values, hence must be input by hand. **MD**: the method of MD algorithm: 1 Verlet; 2 Langevin; 3 Nose-Hoover. Verlet is for NVE, and Langevin and Nose-Hoover are for NVT. Currently, we do not have NPE or NPT MDs. **Mstep**: the number of MD steps. **dt**: the time length for each MD step (in the unit of fs, 1fs = 1 * 10^{-15} s). Note, usually, with H atoms, dt should be 1fs, and with heavier atoms, dt could be 2fs. **Temp1**: the beginning temperature (in K). **Temp2**: the final temperature (in K). Temp2 is only for **MD=**2, 3. During the MD, the program will adjust the temperature, let it goes from Temp1 to Temp2. For Verlet, Temp2 is not used.
NEB_DETAILS	**NEB_DETAIL=imth,nstep,force_tol,** **Nimage,ak,type_string, E0,EN,itype_at2,atom2.config** **imth:1,2,3,** the algorithm used for atomic relaxation. 1: conjugated gradient; 2: BFGS; 3: deepest decent. For NEB calculation, it is better to use 3, otherwise it might not converge. **nstep**: the maximum number of line-minimization steps in the relaxation process.

	force_tol: the force tolerance (eV/A) to stop the relaxation. **Nimage**: the number of images in the NEB method (these are the images except the initial and final two valleys. So, there are in total Nimage+2 configurations in the string of images connection the initial and final configurations. **ak:** the spring constant for the image string (eV/A^2). In the NEB, a string connecting the images are used to ensure the coverage between the initial and final configurations. Ak=0.1 to 1 eV/A2 are reasonable values. Larger ak (especially for type_spring=2), better the convergence, but it can introduce bigger errors (for type_spring=2). **type_spring**: the type of string used in NEB algorithm. 1: the original NEB algorithm (where the string force perpendicular to the string tangent is removed); 2: a conventional string, the perpendicular string force is not removed. Type_spring=2 converges better, but it can introduce an error (larger ak, larger the error). But one can first use larger ak, then after the initial NEB relaxed, re-runs NEB using smaller ak (or type_spring=1). This will help the convergence. **E0,EN**: the precalculated (e.g., using JOB=RELAX) initial (E0) and final (EN) local minima energies (in eV) for configurations in atom.config and atom2.config. Actually, these numbers are not used in the algorithm, but will make plotting more straight forward. **itype_at2**, **atom2.config**: the type of atom2.config file and the atomic position file name: atom2.config. **Itype_at2=1**, atom2.config is the second minimum configuration (the first

	local minimum configuration is given in IN.ATOM=atom.config). Then, from atom.config to atom2.config, Nimage equal distance images will be created by linear interpolations. **Itype_at2=2**, atom2.config contains all the Nimage+2 image configurations (most likely from a previous unconverged NEB run, and copied from MOVEMENT), including the initial and final images. Thus Itype_at2=2 is a continued NEB run following the previous NEB runs. In this case, the atom.config in IN.ATOM=atom.config is not used (but that line still need to be provided). Note, in this case, the atomic positions files for each image inside atom2.config must have first the "Position" section, followed by "Force" section, even though atomic forces are not used. That is the format output in MOVEMENT under JOB=NEB. See NEB.BARRIER for JOB=NEB output file.
NSCALE_VVMD	**NSCALE_VVMD = NSTEP** To scale the kinetic energy in Verlet_MD every NSTEP steps, so the total energy is conserved. The default NSTEP is 100.
RELAX_DETAIL	**RELAX_DETAIL = imth, nstep, force_tol** This is an optional line for "JOB = RELAX". It controls the atomic relaxation steps. Note, in the current atomic relaxation, the unit cell will be kept fixed. **Imth** = 1 (or 2, or 3), indicate the method of relaxation, 1 for conjugated gradient, 2 for BFGS method, 3 for steepest decent (this is mostly for JOB=NEB). The default is imth=1. **Nstep** is the number of relaxation steps (each total energy calculation is one step, i.e.,

	it counts the steps inside the line minimization in the total steps). **Force_tol** (in eV/A) is the force tolerance for the maximal residual force. If the maximum force is less than force_tol, the relaxation will stop. The default values are: "RELAX_DETAIL=1,100,0.01" Note, the JOB = RELAX will output a RELAXSTEPS, and MOVEMENT output files. While RELAXSTEPS gives a summary of the steps, MOVEMENT records the atomic positions for all the steps.
Ecut	The plane wave cutoff energy for wavefunction (in Ryd, note: 1 Ryd = 13.6057 eV). The default value of Ecut is taken from the pseudopotential files atom.upf from its wfc_cutoff value.
Ecut2	The cutoff energy for the soft charge density and the potential (in Ryd). Ideally (for high accurate calculations), Ecut2 should equal 4*Ecut. But in reality, smaller Ecut2 can be used, e.g., 3*Ecut, or 2*Ecut. By default, Ecut2 = 2*Ecut. The n1, n2, n3 are determined by Ecut2. Note, the rho_cutoff value in the pseudopotential files atom.upf is not used.
Ecut2L	The cutoff energy for the hard charge density (in Ryd). Usually, Ecut2L = Ecut2 for norm conserving pseudopotentials, and Ecut2L = 4*Ecut2 for ultra-soft pseudopotentials. Note, the rho_cutoff value in the pseudopotential files atom.upf is not used.
N123	The format is like this: **N123 = n1 n2 n3**

	n1, n2 and n3 are the real space grid to describe the wave function or soft charge density in real space. It is also the FFT grid. The default values are determined by Ecut2 (i.e., make sure the Ecut2 sphere can be held inside the n1,n2,n3 reciprocal box).
NS123	The format is like this: **NS123 = n1s n2s n3s** n1s, n2s and n3s are the real space FFT grid point to calculate the real space projector function. So, these are only used for NONLOCAL=2. For small systems, n1s,n2s,n3s are the same as n1,n2,n3. For large systems, smaller values can be used to save time for projector generation.
N123L	The format is like this: **N123L = n1L n2L n3L** n1L, n2L and n3L are the real space grid for hard charge density. The default values are determined by Ecut2L. For norm conserving pseudopotential, the soft charge equals hard charge, Ecut2L=Ecut2, so n1L, n2L, n3L equal n1, n2, n3. For ultra-soft, Ecut2L= 4*Ecut2, n1L, n2L, n3L = 2*n1, 2*n2, 2*n3.
SPIN	Control the spin of the DFT calculation. **SPIN = 1**, no spin, the default; **SPIN =2**, with spin up and down (but not non-collinear spin. There is no spin-orbit coupling in this version of PWmat).

XcFunctional	Control whether to use GGA calculation. **xcfunctional = 1**, using PBE-GGA **xcfunctional = 0**, use LDA, the default.
COULOMB	Control the Poisson equation solution (for the Coulomb interaction). **COULOMB = 0**, the periodic boundary condition, the default. **COULOMB = 1**, **x1, x2, x3**: the isolated cluster boundary condition. It can avoid the image interaction in this calculation. The x1, x2, x3 (value: 0-1) are the fractional coordination values in the unit cell edge vectors 1, 2, 3, used to cut a box for this special Coulomb solution. In the other word, the center of the box is at: (x1+0.5, x2+0.5, x3+0.5). **COULOMB = 11**, **x1**: A slab calculation along the first direction, with the cut at x1. This can avoid the image interaction between slabs. **COULOMB = 12**, **x2**: A slab calculation along the second direction with the cut at x2. **COULOMB = 13**, **x3**: A slab calculation along the third direction with the cut at x3.
OUT.WG	**OUT.WG = T** (TRUE), PWmat will output a file "OUT.WG", which stores the final wave functions in G-space. When SPIN = 2, an extra file "OUT.WG_2" will also be output. This is the default value.

	OUT.WG = F (FALSE), will not output the wave function file.
IN.WG	**IN.WG = T**, PWmat will read in the initial wave functions in G-space from the file "IN.WG" (e.g., from previous calculation). When SPIN = 2, an extra file "IN.WG_2" will also be read in. **IN.WG = F**, the default, the PWmat will start from random wave function.
OUT.RHO	**OUT.RHO = T**, PWmat will output a file "OUT.RHO", the final charge density in real space grid (n1L, n2L, n3L). This is the default value. If SPIN = 2, PWmat will write out an extra file "OUT.RHO_2". **OUT.RHO = F**, not output the charge file.
IN.RHO	**IN.RHO = T**, PWmat will read in the initial charge density from file "IN.RHO", stored in the real space grid (n1L, n2L, n3L). Note, if both IN.VR and IN.RHO are set to T, the program will use the read-in potential to start the calculation. If SPIN = 2, PWmat will read an extra file "IN.RHO_2". **IN.RHO = F**, not input the charge density. This is the default.
OUT.VR	**OUT.VR = T**, PWmat will output the total potential in file "OUT.VR", stored in real space grid (n1L, n2L, n3L). This is the default value. When SPIN=2, an extra file "OUT.VR_2" will be output.

	OUT.VR = F, not output the potential file.
IN.VR	**IN.VR = T**, PWmat will read in the initial potential from file "IN.VR" in real space grid: (n1L, n2L, n3L). Note, if both IN.VR and IN.RHO are set to T, the program will use the read-in potential to start the calculation. When SPIN=2, an extra file "IN.VR_2" will also be read. **IN.VR = F**, not read in the file. This is the default.
IN.VEXT	**IN.VEXT = T**, PWmat will read in an external potential from file "IN.VEXT" in real space grid (n1L, n2L, n3L). Both the total energy and forces are calculated using this external potential. **IN.VEXT = F**, no external potential is used. This is the default value.
OUT.RHO_SP	**OUT.RHO_SP** = **iflag, kpt1, kpt2, ispin1, ispin2, iw1, iw2** Controls the output of partial charge density (or wave function without square) in real space grid (n1, n2, n3) from selected eigen orbitals within the intervals: k-points: [kpt1, kpt2], spins: [ispin1, ispin2], bands: [iw1, iw2] in the file: "OUT.RHO_SP". This is different from OUT.RHO, since it can select which wave function to be included in the charge density. For: **Iflag = 0**: default, without output. **Iflag = 1**: output partial charge at the end of other

	calculations. **Iflag = 11**: output partial charge using initial input wave functions before doing any other calculations, then stop the code. **Iflag = 2**: output the wave function (instead of the charge density) at the end of other calculations. **Iflag = 22**: output the wave function using initial wave functions before doing any other calculations, then stop the code. For **iflag = 1, 11**, $dens(r) = \sum_i \|\psi_i(r)\|^2$, where index "i" is defined by the above mentioned intervals. For **iflag = 2, 22**, the wave functions are stored one after another in "OUT.RHO_SP" in the sequence: {do ik = kpt1, kpt2; {do is = ispin1, ispin2; {do iw = iw1, iw2; {real part: do ir = 1, n; {Real(psi(ir))}; Imag part: do ir = 1, n; {Imag(psi(ir))};};};}.
OUT.FORCE	**OUT.FORCE = T**, the PWmat will calculate the atomic force, and output the force in file "OUT.FORCE". This is for one shot (one atomic position snap shot) JOB=SCF calculation only. For JOB=RELAX, or JOB=MD, PWmat will always calculate the force. Also note, this will not work for JOB=NONSCF, since there the total energy and SCF charge density will not be calculated. **OUT.FORCE = F**, the default, not calculate the force and output the file.

IN.SYMM	**IN.SYMM = T**, PWmat will use the file "IN.SYMM" (the name is fixed) to perform symmetry operations. The PWmat supports space group symmetry for crystals. "IN.SYMM" should contain space group symmetry operations. "IN.SYMM" is usually generated (together with IN.KPT) by running "check.x" (which will also check whether the IN.SYMM exists if IN.SYMM=T. **IN.SYMM = F**, PWmat will not use any symmetry operations, and not use IN.SYMM. This is the default value.
IN.KPT	**IN.KPT = T**, PWmat will use the k-points from file "IN.KPT" which contains the k-points and their weights. The IN.KPT can be generated (together with IN.SYMM) by running "check.x" with information from variable MP_N123. In band structure calculation, it can be generated by "check.x" together with KP_INV in etot.input. It can also be edited manually. **IN.KPT = F**, PWmat will not use the file "IN.KPT", and it will only use Gamma point. This is the default value.
MP_N123	**MP_N123=nk1 nk2 nk3 sk1 sk2 sk3** This variable is only used by "check.x", not actually used by PWmat. This is the Monkhorst-Pack grids to generate the reduced k-points. When running check.x. If this line is provided, the check.x will generate the IN.SYMM and IN.KPT using the above Monkhorst-Pack parameters, and store them in IN.SYMM, IN.KPT. If no MP_N123, and KP_INV is provided, then when run check.x, no IN.KPT and

	IN.SYMM will be generated. Thus, if don't want to generate k-points, but want to generate symmetry, one needs to set: "MP_N123 = 1 1 1 0 0 0". The sk1, sk2 and sk3 must be either 0 (no offset) or 1 (grid displaced by half a grid point in the corresponding direction).
KP_INV	The number of interval points between two high symmetry k-points. It conflict with "MP_N123", i.e. the user have to use only one of them when necessary. This is for band structure calculation. When check.x is run, it will use the information of the unit cell (edges), generate the special k-points, and link them with lines with KP_INV points. Note KP_INV will also trigger check.x to generate IN.KPT and IN.SYMM.
NUM_ELECTRON	The total number of occupied valence electron in the system. One can use this to make the system charged, or not charged. Note, for charged system calculations, a uniformed back ground charge is used to solve the Possion equation for COULOMB=0. Default value is the value for neutral system.
NUM_BAND	The number of orbitals to be calculated. When SPIN=2, there are NUM_BAND spin-up orbitals and NUM_BAND spin-down orbitals. The default value is about min[1.2*Num_electron/2, Num_electron/2+20].
WG_ERROR	The error tolerance (convergence criterion) for the wave function conjugate gradient iterations (Hartree). The default value is: 10^{-4}. This is related to ALGORITHM0, and ALGORITHM1 lines. It can terminate the CG steps before the

	nline0, nline1 have been reached.
E_ERROR	The error tolerance (convergence criterion) for the total energy (Hartree) in the SCF iterations. The default value is: 2. 10^{-5} (eV). This is related to the SCF_ITER0, SCF_ITER1 lines. It can terminate the SCF iteration before the maximum steps (niter0, niter1) have been reached.
RHO_ERROR	The error tolerance (convergence criterion) using the SCF iteration difference between the input and output charge density. If the relative error of input and output charge density in one SCF step is less than RHO_ERROR, the SCF iteration will be stopped. The default value is 1.5E-6.
W_CG	A variable to control the stopping of the internal CG iterations. This is to estimate the charge density error due to the wave function of CG iteration error. The estimated charge density error should be less than (output-input) SCF charge density error multiplied by w_cg, in order to stop the CG steps. The default value is $7x10^{-2}$. Smaller this value, more stringent requirement, as a result more likely this is not used.
W_SCF	A variable to control the stopping of the SCF iterations. This is to estimate the atomic force error due to the charge density error of SCF iterations. The estimated force error should be smaller than the previous MD or RELAX step force multiplied by W_SCF in order to stop the SCF iterations. Smaller this value, more accurate SCF is used. The default value for JOB = RELAX is 0.003; the default value for JOB=MD is 0.02.

SCF_ITER0	**SCF_ITER0=niter0 nline0** These variables control the charge density self-consistent iterations for the first SCF run for JOB = SCF, RELAX, MD. For RELAX, MD, the first step SCF run (with the initial atomic positions) uses SCF_ITER0, and subsequent steps (for moved atomic positions) uses the values of SCF_ITER1. They are set differently because normally the first run requires much more steps. It is also used for NONSCF run, in which the icmix = 0 for all the niter0 lines in the following. This variable is not used for JOB = DOS. **niter0**: the number of self-consistent (SCF) iterations steps. There must be niter0 lines in "etot.input" following this line in the form of: ALGORITHM0 = imth, icmix, dE, Fermi-Dirac imth, icmix, dE, Fermi-Dirac …….. imth, icmix, dE, Fermi-Dirac See ALGORITHM0 for explanation. The Default value for niter0 is 20. Note the SCF iteration can be stopped before the niter0 has been reached if the E_Error has been satisfied, or the condition specified by w_scf has been reached. So, the stopping of SCF iteration is controlled by four parameters: **niter0, E_error, RHO_error, w_scf**, whichever is satisfied first. **nline0**: the number of CG line minimization steps to solve the

	wave functions according to $H\psi_i = \varepsilon_i \psi_i$ for a given potential (hence H) at each charge self-consistent step. The default value of nline0 is 4. Note, the CG line minimization can be stopped if the error is smaller than WG_ERROR, or the condition specified by w_cg is reached. So, the stopping of CG iterations is controlled by three parameters: **nline0, WG_ERROR, w_cg**, whichever is satisfied first.
Algorithm0	**Algorithm0= imth, icmix, dE, Fermi-Dirac** **imth, icmix, dE, Fermi-Dirac** **……..** **imth, icmix, dE, Fermi-Dirac** These are the niter0 lines immediately following the SCF_ITER0 line. It describes the detail procedures for each SCF step. **imth: 1,** the old band-by-band CG algorithm. It should not be used unless for some special situation. **imth: 3**, the all band conjugate gradient method. This is the default method. **imth: 2**, the DIIS method. This could be faster than imth = 3, but could also have stability problems. It should only be used in SCF iteration steps where the wave function is in some degree converged (e.g., not for random wave functions). **icmix**: 0, no charge mixing and update at this SCF step. In other word, at this step, it is a NONSCF step. For JOB = SCF, RELAX, MD, by default, for the first three SCF steps, icmix =

	0, and icmix = 1 for subsequent steps. For JOB = NONSCF, for all steps, icmix = 0. **icmix: 1**, with charge mixing and update for this SCF step. **dE**: the kT equivalent energy (in eV) for Fermi-Dirac formula to calculate the electron occupations of the eigen wave functions according to their eigen energies ε_i. The default value is 0.1eV. **Fermi-Dirac**: (with possible values: 1, 2, 3, 4, 5). Different formulas for the Fermi-Dirac-equivalent function to calculate the wave function occupation using ε_i and dE. These formulas are: 1, Fermi-Dirac; 2, Gaussian; 3,4,5 Gaussian with other prefactor polynomials. The default value is 1.
SCF_ITER1	**SCF_ITER1 = niter1 nline1** This is for subsequent SCF calculations for JOB = RELAX, MD, except the first SCF step. It has the same meaning as in SCF_ITER0. The default values are niter1=6, nline1=4.
Algorithm1	**Algorithm1= imth, icmix, dE, Fermi-Dirac** **imth, icmix, dE, Fermi-Dirac** **……..** **imth, icmix, dE, Fermi-Dirac** These are the niter1 lines immediately following the SCF_ITER1 line. It has the same meaning as the algorithm0. Usually however, the icmix is always 1. For default, imth = 3, icmix = 1, dE = 0.1, Fermi-Dirac=1.

NONLOCAL	The nonlocal is the nonlocal pseudopotential implementation flag. **nonlocal = 1**, no nonlocal potential. **nonlocal = 2**, the default, real space nonlocal pseudo potential implementation. It used the mask function method. **nonlocal = 3**, G-space nonlocal pseudo potential implementation.
Rcut	The **Rcut** (in Bohr unit, note: 1 Bohr = 0.529177 * 10^{-10} m) is for the cut off radius for nonlocal pseudopotential implementations. It defines the core radius of the nonlocal part. The default value is 3.2 for all calculations except JOB=DOS, for which the default value is 3.8. Larger the Rcut, more accurate, but more expensive.
QijL0_GS	QijL0_GS = 1 (or 2) This parameter is only used by ultra-soft pseudopotential calculations. It controls the core charge implementation in ultra-soft pseudopotential. 1 means the s-component of the core Qij charge is used and implemented in G-space; 2 means the s-component of the core Qij charge is implemented in real-space. The default is 1. We strongly recommend the use of default value since QijL0_GS=2 might introduce some jitter in the energy curve.
QIJ_PD	QIJ_PD = 0 (or 1) It controls whether to include the l=p and d angular momentum core charge density QIJ(r,l) in the ultrasoft

	pseudopotential calculation. The default is 0, which means only the s angular momentum core charge density is used. We strongly recommend the use of 0. The difference is usually very small, but the including of p and d components of the QIJ(r,l) can make the energy manifold unsmooth, thus makes the JOB=RELAX difficult.
PWSCF	PWSCF = T (or F). This parameter controls whether to output pwscf compatible output files (wave function, charge density, and potential), so the result can be run subsequently on pwscf. For T, it will output those files. For F, it will not output. The default is F.
Num_blocked_psi	num_blocked_psi=1/2/3/4/5/…, In choosing this parameter (not 1), PWmat will divide the wavefunctions into num_blocked_psi parts and then put the parts into GPU memory successively one after another during scf iteration. This is to save the use of GPU memory. If a previous run found the GPU out of memory, this can be tried. Larger the num_blocked_psi, smaller is the GPU memory used. Num_blocked_psi=1(default), the program will try to detect the memory situation automatically. If it estimates that there is not enough GPU memory, PWmat will automatically split the wavefunctions into multiple blocks, and to send each block to GPU memory successively. This parameter intends to save the GPU memory to calculate a larger or more complicated systems. So when PWmat tells

	"CUDA Memory insufficent", one can try this parameter by setting num_blocked_psi 2, 3, or 4, and etc. Note that using this parameter will reduce the speed of PWmat (e.g., by a factor of 1.5).
Write2Memory	Write2memory=0, write the wavefunctions into disk. Write2memory=1(default), write the wavefunctions into cpu memory. This parameter is used to save cpu memory to calculate a larger or more complicated systems, in particular for the case multiple k-points are calculated (then one can use Write2memory=0). Note that: it will reduce the performance of PWmat in some degree.

2.2 atom.config

This file describes the supercell box, atomic positions, (optionally) the atomic force, and atomic velocity of the system. It has the following format:

```
-------------------------------------------------------------
64
Lattice vector
   0.1084993850E+02     0.0000000000E+00     0.0000000000E+00
   0.0000000000E+00     0.1084993850E+02     0.0000000000E+00
   0.0000000000E+00     0.0000000000E+00     0.1084993850E+02
Position
   30     0.952534560     0.363594470     0.382027650     1   1   1
   30     0.540553000     0.850230410     0.966359450     1   1   1
   ……
   16     0.242857140     0.140553000     0.684331800     1   1   1
Force
   30    -0.060040948     0.097096690     0.063013193
   30     0.001068674    -0.002521614     0.000147553
   ……
```

16 -0.007955164 -0.008758074 0.029047748

Velocity

30 0.02339881 -0.287387433 -0.109339839

30 -0.23878474 -0.210836551 0.049311111

……

16 0.53761771 -0.023987172 0.288399911

They have the following meanings:

Natom: the number of atoms in the system, as a result, Position, Force, Velocity sections will all have Natom lines, each atom for one atom.

Lattice vector: The header of the lattice vector AL (3, 3) section. There will be three lines following Lattice vector:

AL (1, 1), AL (2, 1), AL (3, 1) (the first vector of the super cell edge in Angstrom)

AL (1, 2), AL (2, 2), AL (3, 2) (the second vector of the super cell edge in Angstrom)

AL (1, 3), AL (2, 3), AL (3, 3) (the third vector of the super cell edge in Angstrom)

Position: the header of the atomic positions of the system. There will be Natom lines following Position, each line describe the position of one atom, in the following form:

Zatom,	x1,	x2,	x3,	imv1,	imv2,	imv3
30	0.2293	0.59822	0.44444	1	1	1

Zatom is the atomic number of this atom, x1, is the fractional coordinate of this atom in the unit of AL(:,1); x2, x3 are the fractional coordinates of this atom in the units of AL(:,2), AL(:,3). Imv1, imv2, imv3 indicates in the atomic relaxation, whether this atom will move in x, y, z (note not the x1, x2, x3) directions. Imv1 (2, 3) = 1, move; 0, not move.

Note the x, y, z coordinates of this atom can be calculated as:

X = AL (1, 1) * x1 + AL (1, 2) * x2 + AL (1, 3) * x3

Y = AL (2, 1) * x1 + AL (2, 2) * x2 + AL (2, 3) * x3

Z = AL (3, 1) * x1 + AL (3, 2) * x2 + AL (3, 3) * x3

Force: The header of the force section. This section is optional. It will be followed by natom lines in the following form:

zatom,	f_x,	f_y	f_z
30	0.0372	0.01112	-0.1021

They are the x, y, z direction atomic forces in eV/Angstrom.

Velocity: The header of the velocity section. This section is optional. If will be followed by natom lines in the following form:

zatom,	v_x,	v_y,	v_z
30	0.39292	-0.222933	0.28211

They are the x, y, z direction atomic velocity in Bohr/fs.

Note, the atom.config file can be converted from other format (.xyz, .xsf) file, using:

">convert_to_config < system.xyz (or xsf)". It will generate a system.config file.

The atom.config file can also be converted into .xyz, .xsf file using

">convert_from_config < xatom.config". It will generate the atom.xyz, and atom.xsf files.

Under JOB=NEB, in NEB_DETAIL line, if itype_at2=1, then atom2.config contains the atomic configuration of the second minimum (for NEB algorithm), while the atom.config in the IN.ATOM=atom.config contains the first minimum atomic configuration, and Nimage image configurations (points) will be created between these two fixed atomic configurations. Note, the atom.config and atom2.config must have the same atom orders. If itype_at2=2, the atom.config in

IN.ATOM=atom.config is no longer used (although it still need to be presented in etot.input), and the atom2.config must contains Nimage+2 configurations (two ends plus Nimage images). Besides, it must also has a fixed form for each configurations: Natom; Lattice; Position; Force; "------" line. In practice, this atom2.config is often copied from MOVEMENT from previous NEB runs.

2.3 Pseudo-potential files: atom.UPF

They are the files copied from the provided libraries. The default Ecut (wfc_cuttoff) is provided in those files. They are ASCII files, thus can be read. In the execution of PWmat, the corresponding pseudopotential files indicated in etot.input must be copied to the running directory. The UPF is the format developed in Quantum Espresso. Currently, PWmat can use norm-conserving pseudopotential, or ultrasoft pseudopotential. There are many groups developing pseudopotentials, and most of them are in the UPF format.

2.4 IN.KPT

The file "IN.KPT" contains the k-point vectors and their weights. It can be generated from preprocessing by running ">check.x". In ">check.x", it will use the Monkhorst-Pack line: "MP_N123=nk1, nk2, nk3, sk1, sk2, sk3" in etot.input, and symmetry of the system to produce the irreducible k-points for total energy calculations. The "IN.KPT" for band structure calculation can also be produced by running ">check.x". The check.x will automatically read "KP_INV" in etot.input and then generates the "IN.KPT".

Note that, for JOB=NEB calculation, very often the image configuration along the path might have lower symmetry than the one at the initial minimum (in IN.ATOM=atom.config). Since symmetry is generated based on atom.config, one must use a general (not highly symmetric) atom.config when generating IN.SYMM and IN.KPT for NEB calculations.

Finally, the IN.KPT can also be edited by hand. It has the following format:

```
-----------------------------------------------------------
2                                                  |   nkpt
2          1.0000                                  |   iflag, a0
0.250      0,250      0.250          0.25         |   ak1, ak2, ak3, weight
0.250      0.250      0.750          0.75
-----------------------------------------------------------
```

Variable	Specification
nkpt	The number of k-points
iflag	**iflag = 1**, the k-points are in x, y, z directions (which is defined by the x, y, z in AL (3, 3) in “atom.input”). **iflag = 2**, the k-points are in the reciprocal lattice of the super cell AL (3, 3). “a0” will not be used.
a0	only used when iflag = 1. (in atomic unit Bohr)
ak1, ak2, ak3	**iflag = 1**, the k-points are defined as: kpx = 2 * π * ak1 / a0 kpy = 2 * π * ak2 / a0

	kpz = 2 * π * ak3 / a0 **iflag = 2**, the k-points are defined as: **k** = **G1***ak1+**G2***ak2+**G3***ak3 Here **G1, G2, G3** are the reciprocal lattice vector of lattice AL (3, 3).
Weight	This is the weight of this reduced k-point (it can represent several symmetric k-points). This is used for SCF calculations, and the total weight as the sum of individual k-points should be 1.

2.5 IN.SYMM

This is the symmetry operation file, usually generated by “check.x”. It contains the space group. It has the following format:

```
-------------------------------------------------------------
12        24        | nsym, nrot
"identity and corresponding fractional translation "
1         0         0
0         1         0
0         0         1
0.000     0.000     0.000
"180 deg rotation - cart. axis [0,0,1] and corresponding fractional translation "
-1        0         0
0         -1        0
0         0         -1
0.000     0.000     -0.500
……
180 deg rotation - cryst. axis [1,1,0]
-1        1         -1
0         1         0
0         0         -1
-------------------------------------------------------------
```

The first line is the two variables: “nsym” and “nrot”. “nsym” is the number of the crystal symmetries operations (space group) and “nrot” is the number of the crystal Bravais lattice symmetries (only for the lattice, not considering the atoms, thus nrot is always larger than nsym). For PWmat, only nsym is used. For the rest of the file, there will be “nsym” operations, each has this following format:

```
-------------------------------------------------------------
"180 deg rotation - cart. axis [0,0,1] and corresponding fractional translation "
-1        0        0                                    !    s (1, 1), s (2, 1),s (3, 1)
0         -1       0                                    !    s (1, 2), s (2, 2),s (3, 2)
0         0        -1                                   ! s (1,3), s (2,3), s (3,3)
0.000     0.000    -0.500                               ! l (1), l (2), l (3)
-------------------------------------------------------------
```

The first line is the explanation of this symmetry operation. The following three line defines the point group rotation matrix s (3, 3) around the origin (x1, x2, x3 = 0, 0, 0, point). The rotation s (3, 3) will convert a real space point (x1, x2, x3) (in lattice cell fractional coordination) into another point following:

y1 = s (1, 1) * x1 + s (2, 1) * x2 + s (3, 1) * x3

y2 = s (1, 2) * x1 + s (2, 2) * x2 + s (3, 2) * x3

y3 = s (1, 3) * x1 + s (2, 3) * x2 + s (3, 3) * x3

The next line defines the fractional translation in the space group. Thus we have:

y1 = y1 + l (1),

y2 = y2 + l (2),

y3 = y3 + l (3).

2.6 Other input files

One might use other input files, e.g., IN.WG (wave function input file), IN.RHO (charge density input file), IN.VR (potential input file), IN.VEXT (external potential file). These are usually generated from the previous SCF calculations (e.g., can be copied over from the corresponding OUT.XXX files). They are binary files. The internal format for wave function file IN.WG is complicated, it is the wave functions in G-space, we usually do not view it, only copy it from OUT.WG to IN.WG for next run. For IN.RHO, IN.VR, IN.VEXT, internally, it has the following format (can be written or read out like this):

Write (in.xx) n1L, n2L, n3L, node1
Write (in.xx) AL
Do iproc = 1, node1
    *Write (in.xx) (vr (ir + nr_nL * (iproc-1)), ir = 1, nr_nL)*
Enddo

Here nr_n = n1L * n2L * n3L / node1. Vr (ii) is a 1D-array form of vr (i, j, k) with ii to (i, j, k) correspondence as: ii = (i-1) * n2L * n3L + (j – 1) * n3L + k.

3. Output files

3.1 Standard output file

Standard (on-screen) output contains the verbose information of each SCF calculation. Standard output has almost all the information for PWmat, but it might be messy to read. If we use: “>mpprun –np num PWmat > out &” to run our job, the standard output will be stored in the file “out”.

3.2REPORT

The “REPORT” file contains the most useful information in a concise way for the run. It might have the following format, with explanation given at the right hand side in blue.

```
               5                1
 IN.ATOM             = 064.config
 JOB                 = RELAX
 RELAX_DETAIL   = 1     100    0.367E-03
 IN.PSP1             = uspp.Zn.txt
 IN.PSP2             = uspp.S.txt
 Ecut                =       21.0000000000000
 Ecut2               =       42.0000000000000
 Ecut2L              =       168.000000000000
 N123                =      45       45       45
 NS123               =      45       45       45
 N123L               =      90       90       90
 SPIN                =    1
 XCFUNTIONA          =     0.000000000000000E+000
 SMTH                =       1.00000000000000
 COULOMB             =       0
 PWSCF               =    F
```

```
IN.WG                =    F
OUT.WG               =    F
IN.RHO               =    F
OUT.RHO              =    F
IN.VR                =    F
OUT.VR               =    F
IN.VEXT              =    F
OUT.RHO_SP           = 0
OUT.FORCE            =    T
IN.SYMM              =    F
IN.KPT               =    F
NUM-Electron         =       576.000000000000
NUM_BAND             =     312
WG_ERRO              =     1.0000000000000000E-004
E_Error              =     5.7600000000000000E-005
w_cg                 =     7.0000000000000001E-002
W_SCF                =     1.0000000000000000E-003
QijL0_GS             =    1
SBFFY                =    1
SCF_ITER0            =       20       4
ALGORITHM0           =       3       0.0000        0.10000       1
                3       0.0000        0.10000       1
                3       0.0000        0.10000       1
                3       1.0000        0.10000       1
                3       1.0000        0.10000       1
                ……..
                3       1.0000        0.10000       1
SCF_ITER1            =        6       4
ALGORITHM1           =       3       1.0000        0.10000       1
                3       1.0000        0.10000       1
                3       1.0000        0.10000       1
                3       1.0000        0.10000       1
                3       1.0000        0.10000       1
                3       1.0000        0.10000       1
NONLOCAL             =    2
Rcut                 =       3.20000004768372
**********************************************
 *********** end of etot.input report ********      ! the above are etot.input.long, can be copied into etot.input,
 **********************************************

 ***** ))) OUTPUT FILE FROM PETOT ((( *******
 for more inf. see file ****: etot.input
 atom config file             ****:      0.000000000000000E+000
 ************************
```

```
recommended n1,n2,n3 from Ecut    ***:     59.8     59.8     59.8
recommended n1,n2,n3 from Ecut2 ***:     42.3     42.3     42.3
Actual          n1,n2,n3 used here    ***:     45       45       45          ! FFT grid determined by Ecut2
recomm.     n1L,n2L,n3Lfrom Ecut2L***:     84.6     84.6     84.6
Actual       n1L,n2L,n3L used here ***:     90       90       90          ! double grid for charge density for uspp
 ************************
nnodes_b=    5    num_group_b=    1    num_group_k=    1          ! nnodes_b=node1, num_group_k=node2, num_group_b=1
natom=         64     ipsp_all=          0                                ! ipsp_all=0, vwr psp; 1, ultrasoft psp
islda=          1         igga= 0.0000000000
Ecut=    21.00       Ecut2=    42.00       Ecut2L= 168.00     Smth=     1.00       ! Smth always=1 in this code.    Ecut in units of Ryd.
totNel=     570.00     mx= 312       tolug=1.0E-03     tolE=1.0E-05     ! totNel: num of electron; mx: the num of calculated ug
ilocal=       2         rcut=     3.20     ntype=         2     ! ilocal: pseudopot. implement; rcut: core cutoff (a.u), ntype: psp. Type.
numkpt=          1    num-sym=          1                ! numkpt: num of k-points;         num-sym: num of symmetry operations
 **********************
 AL1,AL2,AL3 in (x,y,z)
      10.8499385          0.0000000          0.0000000                                        ! in amstrong
       0.0000000         10.8499385          0.0000000
       0.0000000          0.0000000         10.8499385
 ************************
 -------------------------------------------
 iter=     4     ave_lin=    4.0    iCGmth=     3                  ! iter: SCF iter num;    ave_line: CG line min num;
 Ef(eV)                       = 0.3693964E+01                      ! Fermi energy
 err of ug                    = 0.9756E-03                         ! Avery wave function (ug) error: |(H-e)ug|.
 dv_ave, drho_tot         = 0.0000E+00 0.6641E-01
 E_tot                         = -.23469213955264E+04       -.2347E+04
 -------------------------------------------
.............
-------------------------------------------
 iter=    10     ave_lin=    2.0    iCGmth=     3                  ! iCGmth: the method for wave func. Solver: 3, CG, 2: DIIS
 Ef(eV)                       = 0.3896764E+01
 err of ug                    = 0.1108E-03
 dv_ave, drho_tot         = 0.7602E-03 0.4362E-03              ! |V_in-V_out| and    |rho_in-rho_out| errors in SCF (a.u)
 E_tot               = -.23461766740827E+04       0.2133E-04     ! Total energy (in eV), E_thisstep-E_laststep (eV)
 -------------------------------------------------------
 E_Fermi(eV)=     3.89676368689153
 -------------------------------------------------------
 Ef(eV)                       = 0.3896764E+01
 dvE, dvE(n)-dvE(n-1) = 0.5779E-06 0.3464E-06                  ! dvE = \int |V_in-V_out|*rho(r) dr    (a.u)
 dv_ave, drho_tot         = 0.7602E-03 0.4362E-03
 err of ug                    = 0.1108E-03
 -------------------------------------------------------
 Ewald              = -.12541186504745E+04                     ! Ewald energy (eV)
 Alpha              = 0.57532443097826E+02                     ! Pseudopotential Alpha energy (eV)
 E_extV             = 0.00000000000000E+00       0.0000E+00        ! energy due to ext potential: \int V_ext(r)*rho(r) dr (eV)
```

```
 E_NSC              = -.63446274861377E+02       0.2667E+00        ! \sum_i occ(i)*eigen(i),    (eV_
 E[-rho*V_Hxc]= -.10861401146434E+04       -.2666E+00     ! \int V_Hxc(r)*rho(r) dr, V_Hxc: hartree, exchange, correction (eV)
 E_Hxc              = 0.83676461716761E+03       0.2488E+00       ! The sum of Hartree, exchange and correlation energies (eV)
 -TS                = -.40772013136366E-02       -.9536E-04          ! occupation entropy term (eV)
 E_tot(eV)          = -.23461766740827E+04       0.2133E-04          ! total energy, and E_tot(this step)-E_tot(last SCF step) (eV).
 E_tot(Ryd)         = -.17244074436392E+03       0.1567E-05          ! total energy, and E_tot(this step)-E_tot(last SCF step) (Ryd).
 -----------------------------------------------------
Zero temp. E_tot =-.23461746354821E+04 Using formula: E_tot(T)+TS/(N+2), N=       0    ! entropy corrected T=0 energy
 -----------------------------------------------------
 E_Hart,E_xc,E_ion =0.1199722E+04    -.3629580E+03    -.287406E+04       ! E_Hart: Coulomb interaction energy (eV)
 E_rhoVext,E_IVext          =0.000000E+00    0.00000E+00       ! E_rhoVext=\int V_ext*rho*dr,    E_IVext=Ion-V_ext energy (eV)
  E_psiV,E_dDrho =-.3341520719E+03    -.6170243465E+0    ! E_dDrho=sum_i Dj1j2*<beta_j1|psi_i><psi_i|beta_j2>    (eV)
        ave(vtot):v0 =-.3533819079E+00                              ! v0:    (eV).    E_psiV=\int rho*Vtot*dr (eV)
 ave(V_ion_s(or p,d))=ave(V_Hatree)=0; ave(Vtot)=ave(V_xc)=v0
 -----------------------------------------------------
mch_pulay,drho_in,drho_out 0.2559E-04 0.1447E-04              ! |rho_in-rho_out| before and after mch_pulay
 **
 **
 RESULT: atom_move_step, E_tot:       0       -0.234617667408274E+04
 **** finished input atom config calc.    ***
 **** following are atomic    relaxation    ***
 **
 **
 ================================================
 ** atomic relaxation, atom_mov_step:                    1
 ================================================
 ****************************************
 **** within atom_mov_step:                   1
****************************************
 ---------------------------------------------
 iter=     1     ave_lin=    4.0    iCGmth=     3
 Ef(eV)                          = 0.5228815E+01
 err of ug                       = 0.1190E+00
 dv_ave, drho_tot          = 0.1479E+00 0.4236E-01
 E_tot                            = -.23417524427761E+04       -.2342E+04
mch_pulay,drho_in,drho_out 0.3742E+00 0.2839E+00
 ---------------------------------------------
```

3.3RELAXSTEPS

The “relaxsteps” is the file concisely reports the atomic relaxation steps, and the energy and atomic forces at each atom, as well as the SCF convergence for each ab initio step calculations. A typical “RELAXSTEPS” file will look like:

It= 0 NEW E= -0.7526919500493E+03 Av_F= 0.17E+00 M_F= 0.32E+00 dE=.4E-04 dRho=.4E-03 SCF= 4 dL=-.70E-01 p*F =-0.38E-01 p*F0=-0.77E-01 Fch= 0.10E+01

It= 1 CORR E= -0.7527130487491E+03 Av_F= 0.18E+00 M_F= 0.37E+00 dE=.3E-04 dRho=.2E-03 SCF= 3 dL=-.14E+00 p*F=-0.23E-02 p*F0=-0.77E-01 Fch= 0.10E+01

It= 2 NEW E= -0.7527421363137E+03 Av_F= 0.10E+00 M_F= 0.20E+00 dE=.5E-04 dRho=.9E-03 SCF= 2 dL=0.49E-01 p*F =-0.19E-01 p*F0=-0.51E-01 Fch= 0.10E+01

It= 3 CORR E= -0.7527473988358E+03 Av_F= 0.12E+00 M_F= 0.23E+00 dE=.7E-05 dRho=.3E-03 SCF= 2 dL=0.78E-01 p*F= 0.80E-03 p*F0=-0.51E-01 Fch= 0.10E+01

It: The index of total line-minimization number (iteration, or step index)

NEW: this is a new line-minimization direction. The search direction p has changed.

CORR: this is a middle step in the line-minimization process (correction step). Its search direction p is the same as in previous steps (all the way to the last NEW step). Note the energy of this trial step can be higher than previous step. So, to see the convergence, only the energies for the NEW steps should be used.

E : total energy of this step in eV;

Av_F, M_F: average and maximum atomic forces (eV/A);

dE: the SCF iteration E(n)-E(n-1) (eV). This is used to judge whether the SCF iteration is converged. Note, this is not the dE between this relaxation step and previous relaxation step!

dRho: the SCF iteration |rho(n)-rho(n-1)| relative error. This is used to judge whether the SCF is converged.

SCF: the SCF iteration number for this step.

dL: the movement |R-R(new_initial)| of this step (in atomic unit Bohr). R(new_initial)

is the initial atomic position of this line-minimization direction. Note, for the NEW step, there is already one dL. In another word, the dL shown in the NEW line is actually the |R-R(new_initial)| for the R in the following CORR line (i.e, it is the length of the first trial step). Similarly, the dL of one CORR line, is the dL of the R in the following line. The dL's in the NEW, and subsequent CORR (before the next NEW) lines are in the same search direction, and all measured from the beginning point of this new line direction.

p*F: the force of the current step project to the search direction. Note, the purpose of the line-minimization is to make p*F zero (it uses linear interpolation of p*F) to predict the next step size (dL) in this line-minimization.

p*F0: the same as p*F0, however, not use the force of this configuration, but use the force of R(new_initial). Thus this p*F0 is the same throughout one line-minimization direction.

Fch: the force check, calculated as dL*(F+F0)/2 / dE, dL is the displacement for this step from the R(new_initial), F, F0 are the forces are the two ends of this step, (F is the force at the current position, F0 is the initial force at R(new_initial). dE is the total energy difference of this step (the current energy minus the initial energy at the beginning of the new search direction). Fch=1 indicates all the calculations are accurate. Note, for single precision GPU calculation, the dE is usually less accurate than dL*(F+F0)/2, so Fch not equaling 1 can still be fine, since the force is good, and the relaxation algorithms are based on force, not the total energy.

3.4 NEB.BARRIER

The "NEB.BARRIER" is the file concisely report the energies along the images for different relaxation iteration steps. One can yield the barrier height and profiles from NEB.BARRIER. It has the following format:

```
iter=          19    Etot(eV),dist(Bohr),angle(cos(th))
0      -0.75306186045042E+03      0.504486E+00      0.000000E+00
1      -0.75305820517778E+03      0.520270E+00      0.944578E+00
```

2	-0.75304052843358E+03	0.530724E+00	0.846617E+00
3	-0.75296036069356E+03	0.526520E+00	0.355627E+00
4	-0.75266754347227E+03	0.517507E+00	0.883061E+00
5	-0.75234053674623E+03	0.512514E+00	0.961894E+00
6	-0.75234044035416E+03	0.517438E+00	0.961928E+00
7	-0.75266732167841E+03	0.526413E+00	0.883176E+00
8	-0.75296021410969E+03	0.530651E+00	0.356206E+00
9	-0.75304050727950E+03	0.520291E+00	0.846314E+00
10	-0.75305820226225E+03	0.504589E+00	0.944466E+00
11	-0.75306185743092E+03	0.000000E+00	0.000000E+00

This means Nimage=10. The Etot (eV) indicates the total energy of this image. Dist is the distance between the neighboring images (between image and image+1). For good NEB run, the distance should be roughly the same. Angle is the cos(theta) of the angle theta between two R(image+1)-R(image), and R(image)-R(image-1). For good NEB run, cos(theta) should be close to 1 (especially around the barrier height). In practice, it should be fine as long as the cos(theta) is close to 1 near the barrier height. Also, for a good NEB run, the distance between the images should be roughly equal. Iter=19 means this is the 19th line minization result. In NEB.BARRIER, it writes out the results for every relaxation iterations.

3.5 MDSTEPS

The “MDSTEPS” is the file concisely describes the steps of a molecular dynamics simulation. It can have the following format:

Iteration = 8, Etot, Ep, Ek = -0.2346080032E+04 -0.2346382949E+04 0.3029172416E+00
Temperature = 996.40333 dE = -.22E-05 dRho = 0.18E-03 SCF = 2
Iteration = 9, Etot, Ep, Ek = -0.2346080014E+04 -0.2346401749E+04 0.3217343276E+00
Temperature = 1058.29947 dE =-.55E-05 dRho = 0.16E-03 SCF = 2

Etot is the total energy (DFT energy plus kinetic energy) in eV.

Ep is the potential energy (here, DFT energy) in eV.

Ek is the kinetic energy in eV.

For a Verlet algorithm, when everything run well, Etot should be an constant.

Temperature is calculated from the Ek, in Kelvin.

dE is the E(n)-E(n-1) in the SCF iteration (eV).

drho=|rho_in-rho_out| relative error, in the SCF calculation.

SCF is the number of SCF iterations for this MD step.

3.6 MOVEMENT

This is the file generated in RELAX, NEB and MD. It outputs the atom.config of every atomic movement steps (including the correction steps in the line minimization of RELAX) in a single file, one after another. It contains the atomic position, atomic force sections. For MD, it also contains the velocity section. So, it can be copied to atom.config, to continue the run of MD. It can also be converted to other format for visualization (e.g., as animation), by using: ">convert_from_config.x < MOVEMENT".

For JOB=NEB, the MOVEMENT contains the configurations for all the image points.

Note, inside MOVEMENT, the new configuration is appended on the old ones already in the file. The format of MOVEMENT is the same as in atom.config.

3.7 Other output files

There could be other output files, e.g., OUT.WG (wave function output file), OUT.RHO (charge density output file), OUT.VR (potential output file), OUT.DENS (selective wave function charge density file). They are all binary file. They can be copied as IN.XXX file as input for the next run, or they can be visualized. Their

formats are the same as their IN.XXX file (see Section 2.5, Other input files). The format for OUT.DENS is the same as for OUT.RHO. The bpsiiofil10000x file are the wave function to atomic orbital projection file for kpoint x. These are used to generate the partial DOS. It can be removed after DOS is generated.

4. The basic calculations

4.1 Self-consistent calculations (JOB=SCF)

This is a one-shot DFT calculation for a fixed atomic position. It can be used to study the total energy, the magnetic moment, the charge density, the electronic structure, etc. It will not move the atoms. The charge density will be iteratively calculated, until it converges (input charge density equals the output charge density). If subsequent runs (e.g., for NONSCF or DOS) are expected, one should set: OUT.WG = T, OUT.RHO = T, OUT.VR = T (they are all defaults for SCF runs), so they will output files: OUT.WG, OUT.RHO, OUT.VR for subsequent uses. For band structure plot and DOS plot, remember to copy its REPORT to REPORT.SCF (so the Fermi energy in REPORT can be read out). Note, in JOB = SCF calculation, in order to the SCF calculation, the icmix for the late SCF iterations must be 1 (indicating there will be charge mixing and charge update) in the following segment of the etot.input file:

```
-------------------------------------------------------------
SCF_ITER0       =   20      4
ALGORITHM0      =   3       0.0000      0.10000     1
                    3       0.0000      0.10000     1
                    3       0.0000      0.10000     1
                    3       1.0000      0.10000     1
                    Imth,   icmix,        dE,   Fermi-Dirac
-------------------------------------------------------------
```

4.2 None self-consist calculations (JOB=NONSCF)

This is usually used, following a SCF calculation, to study the electronic structure of the system, in particular the band structure. It can use the OUT.VR from the previous calculation (copy them to IN.VR, and set IN.VR to T in etot.input), but with different k-points in IN.KPT, to study the band structure. It can also be used to study some

special cases (e.g., with patched together potential, or charge density). Note, in NONSCF, it can still generate the potential from an input charge density. One needs to set IN.VR=T (to input potential from IN.VR), or IN.RHO=T (to input charge density from IN.RHO, then generate the potential). It will then simply calculate the eigen energies (e.g., for different k-points listed in IN.KPT). The true differences between JOB = SCF and JOB = NONSCF is at the line:

```
------------------------------------------------------------
SCF_ITER0       =   20     4
ALGORITHM0      =   3      0.0000      0.10000      1
                    3      0.0000      0.10000      1
                    3      0.0000      0.10000      1
                    3      0.0000      0.10000      1
                    Imth, icmix,       dE,     Fermi-Dirac
------------------------------------------------------------
```

In SCF, the icmix for the late SCF iteration steps must be 1. (Indicating there will be charge mixing and charge update), while for NONSCF, all the icmix will be zero (no charge mixing and charge update).

To do a band structure calculation, it usually run PWmat in the following procedure: Set Monkhorst-Pack line in etot.input: “MP_N123 = nk1, nk2, nk3, sk1, sk2, sk3”, run “>check.x” to generate IN.KPT, then run a SCF calculation, get OUT.VR, and copy OUT.VR to IN.VR. After that, set “KP_INV” in etot.input (comment out the MP_N123 line), using “>check.x” to generate a different IN.KPT (note, it will overwrite the previous IN.KPT), which contains all the k-points in a band structure path for the typical primary cell shapes (one can also edit IN.KPT by hand), then run a NONSCF calculation using this IN.KPT (IN.KPT=T, IN.VR=T). The band structure information will then be reported in “REPORT”. One can use “plot_band_structure.x” for post-process to view the band structure.

4.3 Density of States Calculations (JOB=DOS)

This usually also follows from one SCF calculation (just like for NONSCF), then to calculate the DOS in this step. In other words, there need to have three calculations in order to get DOS. As for NONSCF, one first gets OUT.VR from SCF calculation. Then in DOS calculation, copy OUT.VR as IN.VR, set IN.VR = T (read this IN.VR). Also, one might want to use more k-points for a nice DOS. To do that, one can use a larger Monkhorst-Pack grids in etot.input (MP_N123 = nk1, nk2, nk3, sk1, sk2, sk3), use ">check.x" to generate a new IN.KPT. Then, one needs to do a JOB = NONSCF calculation to get the eigen energies (stored in EIGEN_ALL) and eigen wave functions (OUT.WG). (If one doesn't want to use more k-points, then one can use the SCF calculation's eigen energies and wave functions, thus skip the JOB = NONSCF calculation step). After this, one can copy OUT.WG to IN.WG, and do a JOB = DOS calculation. The PWmat will output a file: DOS.totalspin (if SPIN=2, there will also be DOS.spinup, DOS.spindown). The format in DOS.totalspin is (each line) (example):

Energy	Total	Zn-s	Zn-p	Zn-d	O-s	O-p	O-d

One can plot this file for graphics. The default energy smoothing/broadening parameter is 0.1 eV. If one wants to have different broadening parameter, or have partial DOS for different atoms, one can use the postprocessing utility code: plot_dos.x. In order to have atom selective partial DOS, one needs to provide a modified atom.config file, with the position section looks like:

---

30 0.952534560 0.363594470 0.382027650 1 1 1 w1
30 0.540553000 0.850230410 0.966359450 1 1 1 w2
……

16 0.242857140 0.140553000 0.684331800 1 1 1 w3

Here w1, w2, w3 are the weights for this atom in the partial DOS. The plot_dos.x also uses bpsiiofil10000x, which are the eigen wavefunciton to atomic orbital projection coefficients for different k-points x, which is output from the JOB = DOS run.

4.4 Atomic Relaxation (JOB=RELAX)

This is to relax the atomic positions following the DFT energy and force. It will generate the RELAXSTEPS, and MOVEMENT files. If it is not fully relaxed, the MOVEMENT can be copied to atom.config (remove all the other iterations, except the last one), then run the PWmat again. Pay attention to w_scf. This parameter is used to stop the SCF iterations. It takes the last iteration average force, and the estimated force error (estimated from |V_in-V_out| in SCF calculations), if the estimated_force_error is less than last_iteration_force*w_scf, then it will jump out the SCF loop. So, smaller w_scf (and larger niter1), more SCF loop might be carried out, and more accurate will be the force. This might be particularly critical if very accurate final atomic positions are needed. The default w_scf for RELAX is 0.003. There are two relaxation methods: imth=1, conjugated gradient method; imth=2, BFGS method (the imth is specified in line RELAX_DETAIL). Their performances are similar. More advanced methods will be introduced in later version of PWmat. When JOB = RELAX is used, one can also include an etot.input line: “RELAX_DETAIL = imth, nstep, force_tol”. If the max_force becomes smaller than force_tol (a.u), the relaxation step will stop (before nstep).

4.5 Nudged Elastic Band Calculations (JOB=NEB)

Nudged Elastic Band (NEB) method is often used to calculate the potential barrier from one local minimum configuration to another local minimum configuration. In order to do NEB calculation, the two local minimum must be known already. They can be calculated by JOB=RELAX, with their atomic configurations being atom1.config, atom2.config, and energies being E1,E2. The idea of NEB is to use a string of images (configuration points) between the two end points (atom1.config, atom2.config). This can guarantee the path can go from one configuration to the other configuration. To avoid the force from the potential (DFT energy) to move the images away from the barrier saddle point (which lower the potential energy), the tangent component of the potential force will be removed. To avoid the force of the elastic string from moving the string away from the saddle point (corner cutting, since an elastic string like to have the minimum length), the perpendicular (to the string tangent) component of the string force will be removed. So the force of the string will only maintain equal distances between the images. This is the essence of the NEB. However, after such modification, there is no guarantee the remaining total force can be written down as a gradient of a potential (i.e, the vortices of the force might not be zero). This can cause significant difficulty in the relaxation procedure to make the force zero (e.g., if one just follows the force to make atomic movement, it is possible that the relaxation iteration can end up in an infinite loop). Another common problem is that string is not smooth, with the vector R(image+1)-R(image) and R(image)-R(image-1) having an angle not close to 180 degree. Thus, there are several points need to be considered when doing a JOB=NEB calculation. First, it is better to use imth=3 (steepest decent for the atomic relaxation method). This is because the CG or BFGS method, which assumes a parabolic potential and the related force might no longer work in NEB. Second, we have implemented two types of string (to deal with the string force). type_string=1 means the original NEB string (remove the perpendicular string force); while type_string=2 means the conventional string (the perpendicular string force is not removed). If the problem failed to converge, one possibility is to first use type_string=2, and a relative large string constant ak. Then, after that is converged, one can do a second NEB calculation (copy MOVEMENT to atom2.config, and use itype_at2=2) using either itype_string=1, or a smaller ak while still use type_string=2. All these choices are to increase the flexibility in NEB calculations.

A normal NEB calculation should have the following steps: (1) using JOB=RELAX to calculate the two local minimum, their atomic positions atom1.config, atom2,config and energies E1, E2. Note, the atomic orders in atom1.config and atom2.config must be the same. (2) Use JOB=NEB, and write NEB_DETAIL as:

IN.ATOM=atom1.config

NEB_DETAIL=imth, nstep, force_tol, Nimage, ak, type_string, E0, EN, itype_at2,

atom2.config

Make sure imth=3, nstep, force_tol can be similar as in RELAX_DETAIL. Nimage is the number of images between the first and last images. Typically Nimage can be 5 to 10. Choose a string constant. Typically, ak=0.1 to 1 (eV/A^2) sounds reasonable. If the relaxation is difficult (to reduce the atomic force, as reported in RELAXSTEPS), one can use type_string=2, otherwise, just use type_string=1. Place the E0,EN from previous RELAX calculations to the line NEB_DETAIL. Use itype_at2=1. Then do the JOB=NEB calculation. If type_string=2 was used, and ak is a bite large, one can do another calculation: (3) copy MOVEMENT into atom.continue (remove all the previous iterations). Replace "1, atom2.config" in NEB_DETAIL by "2, atom.continue". Now, either use type_string=1 (true NEB method), or still use type_string=2, but with a much smaller ak. Do the calculation again. Check "NEB.BARRIER", make sure the images are roughly in equal distance, and the angle between R(image+1)-R(image), and R(image)-R(image-1), especially around the saddle point, is close to 0 degree (costheta close to 1).

4.6 Molecular Dynamics (JOB=MD)

This is for ab initio molecular dynamics simulations. There are three methods: md = 1, Verlet algorithm (for energy conserved true Newton dynamics); md = 2, Langevin dynamics (with a viscosity and a thermos bath, for given temperature control simulation); md = 3, Nose-Hoover (stochastic methods for given temperature control simulation). We recommend to use either md=1, or md=3. This simulation will output MDSTEPS, and MOVEMENT. One can continue the simulation by copying MOVEMENT to atom.config (remove all the other iterations except the last one), then restart the calculation (only retype the running commands in the terminal). The PWmat will automatically detect whether there is a velocity section in atom.config. If yes, then it will use it as the initial velocity. If no, it will use temp1 to randomly

generate an initial velocity. When using “JOB = MD”, one has to include an etot.input line: “MD_DETAIL = md, mstep, dt, temp1, temp2”.

“md” specifies the method. For md=1 (verlet), only when it is start from scratch, temp1 will be used. It is used to generate the initial random velocity according to this temperature. For md=1, temp2 is not used. For continued run, the initial velocity is read-in from the atom.config file, so temp1 is not used. For md=2, 3, if start from scratch (initial atom.config does not provide the velocity), then temp1 is used to generate the initial velocity. The md=2,3 dynamics will scale the temperature linearly with steps from temp1 to temp2.

4.7 Compatible runs with PWSCF

The PWmat can be run compatibly with the open source code PWSCF. Mostly, the PWmat can generate the wave function, charge density, and potential files, which can be read by the PWSCF program, or the PWSCF compatible programs (e.g., Wannier90 function generator, or GW calculations). These programs can be run on CPU. To run those programs, the user is responsible to prepare their control input files. One should also copy the corresponding PWSCF pseudopotential files from our library (they are the same vwr.atom, uspp.atom.txt potentials, but in the PWSCF pseudopotential format). Note that, due to slightly different implementation of the nonlocal pseudopotentials, the PWSCF, and PWmat energies and forces might be slightly different. But the difference is rather small.

With the compatibility of PWmat and PWSCF, one can fully take the advantages of the wide functionalities of the PWSCF, while enjoy the speed of PWmat for some of the key calculations. The available PWSCF capabilities include: Wannier function

generation; linear-response phonon band structure calculation; linear-response TDDFT calculation; GW calculation. We refer the user to consult the open source PWSCF manual for how to calculate these properties.

To generate the PWSCF compatible wave function and potential files, set PWSCF=T in etot.input.

5. Pre- and post-processing programs

5.1 Pre-processing

We provide the utility programs to help the preparing of the PWmat input files.

5.1.1 convert_to_config.x

One useful program is “convert_to_config.x”, which convert the atomic position file from other formats (.cell, .xsf, .vasp) to .config file format which we use. We also provide a tutorial for how to generate .cell, .xsf or .vasp file from visualization tools, or online database.

5.1.2 check.x

Another major preprocessing tool is “check.x”, it will check the etot.input file, to see whether there is any error in it (e.g., not missing any line), it will generate a etot.input.long, spelling out all the default parameters to be used in the calculation. If wished, one can copy etot.input.long into etot.input, manually change some parameters. One can also use the original etot.input for PWmat runs. Although not necessary, we do strongly encourage the user to use check.x for the first time to calculate one system. One important thing is that “check.x” will spell out the FFT grid n1,n2,n3. That will help the user to choose node1 (first line in etot.input), which must evenly divide n1*n2. So, one might want to change node1 in etot.input according to n1,n2,n3, or change n1,n2,n3 according to node1.

Another thing one has to use check.x is to generate IN.KPT, IN.SYMM if one wants to use k-points and symmetry. The Monkhorst-Pack parameters: “MP_N123 = nk1, nk2, nk3, sk1, sk2, sk3” or alternatively, the band structure parameter: KP_INV=m, must be included in etot.input, in order for check.x to generate IN.KPT, IN.SYMM. IN.KPT = T, IN.SYMM = T must be used in order to use these k-points, and symmetry file. Thus, in other words, in order to use k-points, and symmetry point, one must run “>check.x” first. Note, IN.KPT, and IN.SYMM should be used together.

Another useful function of check.x is to generates the IN.KPT for band structure calculations. After setting the “KP_INV” in etot.input, it will take the information of the “atom.config” file, match that to one of the 14 primary cell shapes, then generate the corresponding high symmetry k-points, and connect them into a band structure path. The resulting k-points will be written in IN.KPT. This can be used in the JOB = NONSCF to generate the band structure.

5.2 Post-processing

The PWmat package also provides post-processing programs to facilitate the post processing of the runs. Mainly, it has the following post-processing programs:

5.2.1 convert_from_config.x

This program will convert the xatom.config file, or MOVEMENT into .xyz and .xsf formats, so they can be used by different visualization tools for viewing. We provide a

tutorial for how to use these visualization tools. This is the sister program of "convert_to_config.x" in the pre-processing.

5.2.2 plot_band_structure.x

Before run the plot_band_structure.x, please prepare OUT.EIGEN, IN.KPT and REPORT.SCF (this file is copied the SCF calculation output REPORT into REPORT.SCF). Then it will generate the following files: band.eps, band.jpb and origin.dat (the data file of band structure), which can be used to plot band with specified scale and regions.

5.2.3 plot_dos.x

This program uses the DOS.totalspin with 0.1 eV energy broadening. If one want to shift the Fermi energy to zero, the REPORT.SCF is needed. After running the plot_dos.x, it will generate dos.eps or dos.jpg. However, if one wishes to change this broadening, or more importantly, plot the atomic selected partial DOS, one can use this plot_DOS_new.x. To do that, one needs to provide a modified xatom.input file, with the following format:

```
-------------------------------------------------------------
30      0.952534560     0.363594470     0.382027650     1   1   1    w1
30      0.540553000     0.850230410     0.966359450     1   1   1    w2
……….
16      0.242857140     0.140553000     0.684331800     1   1   1    w3
-------------------------------------------------------------
```

Here w1, w2, w3 are the weight for this atom in the partial DOS.

5.2.4 Convert_rho.x

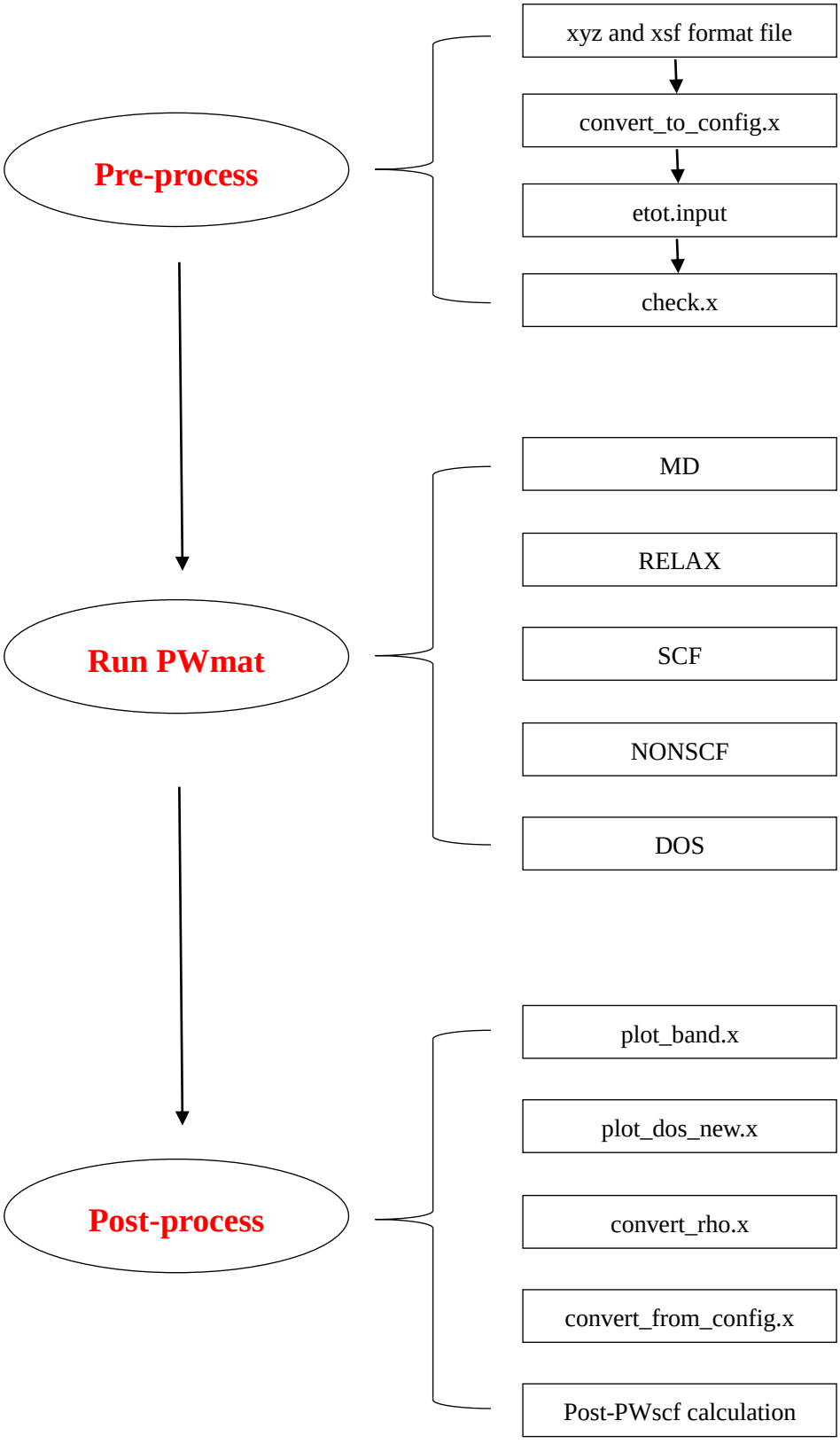
This program convert the potential, or charge density file (OUT.VR, OUT.RHO. OUT.RHO_SP) to a rho.xsf format, so they can be view by VESTA. To use it, type:

">convert_rho.x OUT.RHO".

It will generate RHO.xsf. Note in above, do not use: ">convert_rho.x < OUT.RHO".

Appendix: A

WORKFLOW



Appendix: B

Useful Websites

1. PEtot:

   http://cmsn.lbl.gov/html/PEtot/PEtot.html

2. Quantum Espresso:

   http://www.quantum-espresso.org/

3. American Mineralogist Crystal Structure Database:

   http://rruff.geo.arizona.edu/AMS/amcsd.php

4. NIST Chemistry WebBook:

   http://webbook.nist.gov/chemistry/

5. XCyrSDen:

   http://www.xcrysden.org/

6. VMD:

   http://www.ks.uiuc.edu/Research/vmd/

7. Jmol:

   http://jmol.sourceforge.net/

8. Nvidia Cuda Zone:

   https://cudazone.nvidia.cn/